Virginia Tech ❖ Bradley Department of Electrical and Computer Engineering

ECE-4984 SS: Certified Programming / ECE-5984 SS: Adv Certified Programming
Fall 2017, Course Syllabus

Course Reference Numbers (CRNs)

- **Physical presence on VT Blacksburg campus:**
  - ECE 4984 - Certified Programming: **89530**
  - ECE 5984 - Advanced Certified Programming: **89528**
- **Off-campus online through WebEx:**
  - ECE 5984 - Advanced Certified Programming: **89536**

Instructor

Dr. Peter Lammich
Research Assistant Professor
ECE Dept., Virginia Tech
Office: Durham 352, Blacksburg, VA 24061

Instructor office hours information

- *Date and time*:
  - Mon, 4:00pm – 5:00pm;
- *Location*: Durham 352
- *Phone*: TBA
- *E-mail*: `lpeter1@vt.edu`

Course objectives

Most computer programs of non-trivial complexity contain bugs. Depending on the area of application, the effect of a bug may be from harmless to catastrophic. Computer programs are usually verified by testing, which is notoriously incomplete, as it can never cover all (infinitely many) possible inputs for a program.

A higher level of trust can be achieved by formal verification of a program, i.e., mathematically proving that the program behaves as expected. In order to do so, one has to mathematically define the meaning (semantics) of a program, and to provide a mathematical description of the desired behavior (specification).

However, mathematical proofs, in particular those for computer program correctness, tend to get large, unreadable, and error-prone if executed with pen and paper. To this end, interactive theorem provers (ITPs) can help: ITPs are computer programs that check mathematical proofs to be valid, and help the user in creating mathematical proofs. Many ITPs are designed to reduce potential bugs to a small logical inference kernel, which is well tested. Thus, a proof that is checked by an ITP is very likely to be correct.

This course introduces the state-of-the-art interactive theorem prover Isabelle/HOL and its application to verification of simple computer programs. The main objective is to teach the students to use ITPs on their own, thus a main focus will be on hands-on exercises and practical homework.

Upon completion of the course, the student will:

- be able to write precise formal proofs with the theorem prover Isabelle,
- be familiar with operational semantics of some simple imperative program constructs, and
- be able to verify simple imperative programs with Isabelle.

Prerequisites

- *4984*: ECE 2574 Introduction to Data Structures and Algorithms
- *5984*: Graduate standing
- *Both levels*: Experience with imperative programming language (C, C++, Java,...)

Course meeting time and location

Meeting time: Mondays and Wednesdays, 5:30PM-6:45PM
Location: Torgersen 1050.

Required and recommended texts

Required

- Nipkow, T. and Klein, G. (2014). *Concrete Semantics*. Springer.

Recommended:

- Nipkow, T. *Programming and Proving in Isabelle/HOL* (`http://isabelle.in.tum.de/dist/Isabelle2016-1/doc/prog-prove.pdf`)

Development environment

The students have to bring a laptop with the most recent stable version of Isabelle/HOL to the course. Isabelle/HOL can be obtained from `http://isabelle.in.tum.de/`.

Should the above points concerning the development environment be an issue, the student is encouraged to contact the instructor.

Course Website

The website of the course is in Canvas: `https://canvas.vt.edu/`. Use your Virginia Tech PID and password to access the web site. The website will be used to distribute lecture transparencies, code examples, homework assignments, and additional readings, provide solutions after exams and assignments, and announce exam times and any modifications to the syllabus. All students are expected to have access to electronic mail and the Web.

Homeworks, Projects, and Exams (Tentative and subject to change)

Both Sections

- Approx. 10 homework assignments (the actual number of assignments maybe higher or lower and will depend on many factors including class progress, topical dependencies, scheduling issues, etc)
- Bonus homework assignments (usually add-ons to regular homework, infrequent): When computing your percentage, points achieved for bonus homework count on your side, but bonus homework is not considered for the maximum achievable points.

Graduate Section

- 1 project assignment (3 or 4 weeks)
- Final exam, take-home (put on canvas on 7pm, Dec 15. You have 24h to solve it and submit solution.)
- Grading weights: 60% homework (all assignments weigh equal), 25% project, 15% final exam

Undergraduate Section

- 1 project assignment (2 weeks)
- Final exam, in-class, Dec 15, 7pm–9pm, Torgerson 1050. You may bring in two **handwritten** sheets (legal, letter, or smaller).

- Grading weights: 70% homework (all assignments weigh equal), 15% project, 15% final exam

Grading:

Semester grades will be determined after all course work is completed and graded. From your homework, project, and exam points, we will compute a cumulative score between 0 and 100, according to the weights specified above. If, due to bonus points, your cumulative score is higher than 100, it will be capped at 100. (and you'll get the highest possible grade anyway)

The mapping from the cumulative score to letter grades will be the same for each section (undergraduate, graduate), but it will not be based on a fixed, predetermined curve or point range. If you have questions or concerns about your performance or grades, please discuss them with the instructor.

Important Grading Policies

All assignments (i.e., projects, homeworks, take-home exam) must be submitted before the announced due date/time and must be turned in electronically (details TBA on first homework assignment). Homeworks submitted after that time will not be accepted unless extenuating circumstances exist and arrangements are made prior to the due date. Also, no makeup exams will be given. The only exceptions are medical emergencies or other extraordinary circumstances that must be justified with proper documentation.

If you feel that an error is made in grading a homework or an exam, you must present a written appeal within one week after the graded work is returned to you (appeal by e-mail is sufficient). Verbal appeals are not allowed and grades will not be changed after the one-week period. Your appeal should be specific. Submit your appeal to the GTA for homeworks and to the instructor for exams.

Honor Code Policy

Adherence to Virginia Tech's honor code (Undergraduate Honor System: `http://www.honorsystem.vt.edu/`) is expected in all phases of this course. All graded work is expected to be the original work of the individual student unless otherwise directed by the instructor.

In working on homeworks, discussion and cooperative learning are allowed and, in fact, encouraged. However, copying or otherwise using another person's solutions to the homework problems is an honor code violation. Individual work is to be the work of the individual student. You may discuss general concepts, such as software libraries, Internet resources, or class and text topics, with others. However, any discussion or copying of homework solutions, specific code, or detailed report content is an honor code violation. All source material used in homework code and reports must be properly cited. If you are using a shared computer or disk, it is an honor code violation to leave your source, report, or other files on the computer where others may access them, and it is an honor code violation to access other students' files.

The Undergraduate Honor Code pledge that each member of the university community agrees to abide by states: âĂIJAs a Hokie, I will conduct myself with honor and integrity at all times. I will not lie, cheat, or steal, nor will I accept the actions of those who do.âĂİ

Students enrolled in this course are responsible for abiding by the Honor Code. A student who has doubts about how the Honor Code applies to any assignment is responsible for obtaining specific guidance from the course instructor before submitting the assignment for evaluation. Ignorance of the rules does not exclude any member of the University community from the requirements and expectations of the Honor Code.

For additional information about the Honor Code, please visit: `https://www.honorsystem.vt.edu/`

Students are expected to not use any tutoring service, on-campus or off-campus, paid or unpaid, toward completing the course homeworks. If you plan to use any tutoring service, seek permission from the instructor before hand.

Please discuss any concerns about the honor code or any questions that you may have about what is or is not permitted with the instructor.

Any violations of the honor code will automatically be forwarded to the Office of the Honor System.

Special Needs or Circumstances

Any student with special needs or circumstances should feel free to meet with or otherwise contact the instructor.

- *Disability accommodation*: Reasonable accommodations are available for students who have documentation of a disability from a qualified professional. Students should work through Virginia Tech's Services for Students with Disabilities (SSD). Any student with accommodations through the SSD Office should contact the instructor during the first two weeks of the semester;
- *Religious accommodation*: If participation in some part of this class conflicts with your observation of specific religious holidays during the semester, please contact the instructor during the first two weeks of class to make alternative arrangements;
- *Accommodations for medical or personal/family emergencies*: If you miss class due to illness, especially in the case of an exam or some deadline, see a professional in Schiffert Health Center. If deemed appropriate, documentation of your illness will be sent to the Dean's Office for distribution to the instructor. If you experience a personal or family emergency that necessitates missing class, contact the Dean of Students.

Course topic outline (tentative)

The course will roughly follow the structure of the Concrete Semantics book (`http://www.concrete-semantics.org/`), covering chapters 2-8, parts of 9, and 12:

Intro to Isabelle/HOL Introducing functional programming in Isabelle/HOL. This part will teach how to use Isabelle/HOL as an proof enhanced functional programming language. It introduces the basic techniques of functional programming in HOL (algebraic datatypes, pattern matching, recursive functions), together with the corresponding proof techniques (rewriting and structural induction). Examples are motivated by the application to programming language semantics. (e.g. expressions over integers and their compilation to simple stack and register machines.)

Advanced Isabelle/HOL This part covers the more advanced features of Isabelle/HOL that are necessary for non-trivial proofs. We introduce the concepts of inductive data types and the corresponding proof techniques. Moreover, we introduce the structured proof language Isar, that allows one to write human-readable proofs. Examples in this section aim at illustrating the proof techniques using simple and intuitive examples. (e.g. odd/even numbers, reflexive-transitive closure)

Semantics of Imperative Languages In this part, Isabelle/HOL is used to formalize the semantics of a simple imperative programming language. It covers operational semantics (small-step and big-step) and related concepts (e.g. flow-graphs).

Verified Compilation In this part, Isabelle/HOL is used to verify a compiler for the simple imperative language to a simple stack machine. (register machine)

Program Verification and Hoare Logic This part teaches verification of imperative programs with Isabelle/HOL. It introduces axiomatic semantics (Hoare-Logic), shows its soundness and completeness, and covers verification condition generation. Examples include the actual verification of simple programs (e.g. Euclid's algorithm, square-root by bisection)